USB Flash Disk and SD Card High Speed File Management Control Chip CH378

Datasheet (II): Auxiliary Commands and USB Basic Transmission Commands

Version: 4B

http://wch.cn

1. Additional Commands

Code	Command name CMD_	Input data	Output data	Command purpose
The following commands are auxiliary commands related to USB host mode				
07H	BULK_WR_TEST	Input length (2) Data stream (N)		Batch data write tests
08H	BULK_RD_TEST	Input length (2)	Data stream (N)	Batch data read tests
0AH	READ_VAR8	Variable address	Data	Read the specified 8-bit CH378 system variables
0BH	WRITE_VAR8	Variable address Data		Set the specified 8-bit CH378 system variables
0CH	READ_VAR32	Variable address	Data (4)	Read the specified 32-bit CH378 system variables
0DH	WRITE_VAR32	Variable address Data (4)		Set the specified 32-bit CH378 system variables
0EH	GET_REAL_LEN		Data (4)	Quickly return the actual length of the last command execution
16H	TEST_CONNECT		Connection status	Check the connection status of USB device or SD card
25H	DIRTY_BUFFER			Clear internal disk and file buffers
41H	CLR_STALL	Endpoint number	Generate interrupt	Control transmission: Clear endpoint error
45H	SET_ADDRESS	Address value	Generate interrupt	Control transmission: Set USB address
46H	GET_DESCR	Descriptor type	Generate interrupt	Control transmission: Get the descriptor
49H	SET_CONFIG	Configuration Value	Generate interrupt	Control transmission: Set USB configuration
4DH	AUTO_SETUP		Generate interrupt	Automatically configure USB device
4EH	ISSUE_CTRL_TRAN		Generate interrupt	Execute control transmission
51H	DISK_INIT		Generate interrupt	Initialize USB memory
52H	DISK_RESET		Generate interrupt	Reset USB storage device
53H	DISK_SIZE		Generate interrupt	Get the capacity of the USB memory

58H	DISK_INQUIRY		Generate interrupt	Query USB memory features
59H	DISK_READY		Generate interrupt	Check USB memory readiness
5AH	DISK_R_SENSE		Generate interrupt	Check USB memory error
5DH	DISK_MAX_LUN		Generate interrupt	Get the maximum cell number of the USB memory device
60H	SET_LONG_FILE_NAME	String ending in 2 zeros		Set the filename of the long file to be operated
61H	GET_LONG_FILE_NAME		Generate interrupt	Get the corresponding long filename from the full short filename path
62H	LONG_FILE_CREATE		Generate interrupt	Create a long filename file
63H	LONG_DIR_CREATE		Generate interrupt	Create a long file name directory and open it
64H	AUTO_DEMO		Generate interrupt	Automatic chip demonstration
65H	GET_SHORT_FILE_NAME	String ending in 2 zeros	Generate interrupt	Get the corresponding short filename path by the long filename
66H	LONG_FILE_OPEN	String ending in 2 zeros	Generate interrupt	Open the file by the long filename
67H	LONG_FILE_ERASE	String ending in 2 zeros	Generate interrupt	Delete the file by the long filename
The following commands are commands related to USB device mode				
11H	SET_USB_ID	VID low bytes VID high bytes PID low bytes PID high bytes		Set the USB vendor ID (VID) and product ID (PID)
13H	SET_USB_SPEED	USB speed value		Set the current USB device speed
14H	GET_USB_SPEED		USB speed value	Get the current USB device speed
17H	INIT_ENDPx	Endpoint index Endpoint number Endpoint type Endpoint direction The endpoint package size is at high bytes The endpoint package size is at low bytes Interrupt status code		Initialize USB endpoint (External firmware)
18H	SET_INDEXx_IN	Endpoint index working mode		Set the working mode of IN endpoint

19H	SET_INDEXx_OUT	Endpoint index working mode		Set the working mode of OUT endpoint
23H	UNLOCK_USB	Endpoint index		Release the current USB buffer
29H	RD_USB_DATA	Endpoint index	Data Length Data stream	Read the data block from the specified USB endpoint buffer and release the buffer
2AH	WR_USB_DATA	Endpoint index Data Length Data stream		Write the data block to the send buffer of the specified USB endpoint and send the data
2BH	WR_USB_DATA0	Data Length Data stream		Write the data block to the send buffer of the USB control endpoint and send the data

1.1. CMD_BULK_WR_TEST

This command is used for chip batch data write tests. Firstly write 2-byte data length, and then input subsequent data streams one by one by length.

1.2. CMD_BULK_RD_TEST

This command is used for chip batch data read tests. Firstly write 2-byte data length, and then read subsequent data streams one by one by length. This command is generally used in conjunction with the command CMD_BULK_WR_TEST to test the correctness when the external system and chip carry out bulk data transmission.

1.3. CMD_READ_VAR8

This command is used to read the specified 8-bit (single byte) file system variables. This command requires to input 1 byte of specified variable address and output the data for this variable.

1.4. CMD_WRITE_VAR8

This command is used to set the specified 8-bit (single byte) file system variables. This command requires to input two bytes of data, respectively the specified variable address and the specified variable data.

1.5. CMD_READ_VAR32

This command is used to read the specified 32-bit (4 bytes) file system variables. This command requires to input 1 byte of specified variable address and output the data for this variable. The variable data has a total of 4 bytes, which are the lowest byte of data, the lower byte of data, the higher byte of data, and the highest byte of data.

1.6. CMD_WRITE_VAR32

This command is used to set the specified 32-bit (4 bytes) file system variables. This command requires to input 5 data, respectively specified variable address, the lowest byte of variable data, lower byte of data, higher byte of data and the highest byte of data.

1.7. CMD_GET_REAL_LEN

This command is used to quickly return the actual length after the last command execution. Output 4 bytes of data, which are the lowest byte of the actual length, the lower byte of the actual length, the higher byte of the actual length, and the highest byte of the actual length in sequence. For example, after the command CMD_BYTE_READ or CMD_SEC_READ is successfully sent and an interrupt is generated, the command can be sent to get the number of bytes or sectors actually read.

1.8. CMD_TEST_CONNECT

This command is used to query the connection status of the current USB device or SD card. After completion, USB_INT_CONNECT, USB_INT_DISCONNECT, or USB_INT_USB_READY is output. USB_INT_CONNECT indicates that the USB device is just connected or has been connected but has not been initialized. USB_INT_DISCONNECT indicates that the USB device has not been connected or has been disconnected. Status USB_INT_USB_READY indicates that USB device has been connected and initialized (USB address has been assigned). 0 indicates that the command is not completed and the status can be read later.

1.9. CMD_DIRTY_BUFFER

This command is used to clear internal disk and file buffers in the host file mode. After being in host file mode, CH378 always stores some frequently used data in the internal disk buffer, but when some commands (such as CMD_RD_DISK_SEC or CMD_WR_DISK_SEC, etc.) are executed, the buffer will be used, causing the buffer data to be invalid. In order to prevent CH378 from misusing the invalid data, it is necessary to notify CH378 to clear the internal buffer after other commands use the internal buffer.

1.10. CMD_CLR_STALL

This command is used to clear the wrong control transmission command of the endpoint. This command requires to input 1 byte of data to specify the endpoint number of the USB device for which errors will be cleared. Valid values are 01H ~ 0FH for OUT endpoint and 81H ~ 8FH for IN endpoint. This command is used to simplify the standard USB request CLEAR_FEATURE. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully.

1.11. CMD_SET_ADDRESS

This command is used to set the control transmission command of the USB address. This command requires to input 1 byte of data to specify the new address USB device, and the valid addresses are 00H-7FH. This command is used to simplify the standard USB request SET_ADDRESS. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully.

1.12. CMD_GET_DESCR

This command is used to get the control transmission command of the descriptor. This command requires to input 1 byte of data to specify the type of descriptor to be obtained. The valid type is 1 or 2, corresponding to DEVICE descriptor and CONFIGURATION descriptor. CONFIGURATION descriptor also includes the interface descriptor and the endpoint descriptor. This command is used to simplify the standard USB request GET_DESCRIPTOR. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully. MCU can get the descriptor data through the command CMD_RD_HOST_REQ_DATA.

1.13. CMD_SET_CONFIG

This command is used to set the control transmission command of the USB configuration. This command requires to input 1 byte of data to specify a new USB configuration value. If the configuration value is 0, the configuration will be canceled, otherwise it shall be taken from the configuration descriptor of the USB device. This command is used to simplify the standard USB request SET_CONFIGURATION. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully.

1.14. CMD_AUTO_SETUP

This command is used to automatically configure USB device and does not support SD card. This command

is used to simplify the initialization steps for the common USB device and is equivalent to the sequence of multiple commands such as GET_DESCR, SET_ADDRESS, SET_CONFIGURATION and so on. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully.

1.15. CMD_ISSUE_CTRL_TRAN

This command enables CH378 to execute USB control transmission. Before this command is initiated, the 8-byte SETUP package must be written to the internal buffer through the command CMD_WR_HOST_OFS_DATA. CH378 requests an interrupt from MCU after the command is executed. MCU can read the interrupt status as the operation status of the command. If the operation status is ERR_SUCCESS, the command will be executed successfully; otherwise, the command will be executed unsuccessfully. MCU can further analyze the cause of failure according to the operation status.

For OUT transaction sending data, the subsequent data packet to be sent shall be written into the internal buffer through the command CMD_WR_HOST_OFS_DATA, followed by the 8-byte SETUP package. For IN transaction receiving data, the received data shall be read through the command CMD_RD_HOST_REQ_DATA after successful execution.

1.16. CMD_DISK_INIT

This command is used to initialize USB storage device and does not support SD card. The USB device shall be enumerated before this command is executed. If the device is a USB storage device, the USB storage device can be initialized directly through this command. After successful initialization, read and write operations can be performed for the USB storage device. CH378 requests an interrupt from MCU after the command is executed. If the USB device has been disconnected, the interrupt status may be USB_INT_DISCONNECT. If the USB device is not recognized or the USB storage device is not supported, the interrupt status will be usually USB_INT_DISK_ERR. If the USB storage device is initialized successfully, the interrupt status will be ERR_SUCCESS.

1.17. CMD_DISK_RESET

This command is used to reset USB storage device through the control transmission and does not support SD card. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the command will be executed successfully.

The complete reset process includes: resetting the USB storage device through this command, resetting Bulk-IN endpoint through the command CLR_STALL, and resetting Bulk-OUT endpoint through the command CLR_STALL.

When an error occurs on the USB storage device, CH378 will analyze the cause of the error and automatically select whether the USB device is reset or not as required.

1.18. CMD_DISK_SIZE

This command is used to get the capacity of the USB storage device and does not support SD card. After successfully initializing the USB storage device, this command gets the total capacity of the USB storage device. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the data can be gotten through the command CMD_RD_HOST_REQ_DATA. The data is usually 8 bytes. The first 4 bytes constitute double-word data with high bytes in the front, which is the total number of sectors of USB storage device. The last 4 bytes constitute the double-word data with high bytes in the front, which is the number of bytes of each sector. The result of multiplying two data is the total capacity of USB storage device in bytes.

1.19. CMD_DISK_INQUIRY

This command is used to inquire the features of the USB storage device and does not support SD card. CH378 requests an interrupt from MCU after the command is executed. If the interrupt state is ERR_SUCCESS, the data can be gotten through the command CMD_RD_HOST_REQ_DATA. The data is usually 36 bytes, including the features of USB storage device and the identification information of vendor and product. This command is generally not needed unless a new logical unit is analyzed.

1.20. CMD_DISK_READY

This command is used to check whether the USB storage device is ready and does not support SD card. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, it will indicate that the USB storage device has been ready.

1.21. CMD_DISK_R_SENSE

This command is used to check the error of the USB storage device. CH378 requests an interrupt from MCU after the command is executed. The interrupt status is normally ERR_SUCCESS, and the error can be analyzed after the data is gotten through the command CMD_RD_HOST_REQ_DATA.

1.22. CMD_DISK_MAX_LUN

This command is used to get the maximum logical unit number of the USB storage device through control transmission. CH378 requests an interrupt from MCU after the command is executed. If the interrupt state is ERR_SUCCESS, the data can be gotten through the command CMD_RD_HOST_REQ_DATA. The data is usually 1 byte.

1.23. CMD_SET_LONG_FILE_NAME

This command is used to set the filename of the long file to be operated or the directory name. The input data is a string ending in 2 zeros, not exceeding 520 bytes in length.

1.24. CMD_GET_LONG_FILE_NAME

This command is used to get the corresponding long filename from the full short filename path (which can be a file or folder). Before this command is initiated, it is necessary to set the full pathname of the short filename through the command CMD_SET_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the data can be gotten through the command CMD_RD_HOST_REQ_DATA. The output data includes 2-byte long filename total length (max. 520 bytes) and subsequent long filename characters in sequence.

1.25. CMD_LONG_FILE_CREATE

This command is used to create a long filename file. If the file already exists, delete it before creating. Before this command is initiated, it is necessary to set the full pathname of the short filename through the command CMD_SET_FILE_NAME and then set the corresponding long filename through the command CMD_SET_LONG_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the long filename file will be created successfully.

1.26. CMD_LONG_DIR_CREATE

This command is used to create a long filename directory. If the directory already exists, delete it before creating. Before this command is initiated, it is necessary to set the full pathname of the short directory name through the command CMD_SET_FILE_NAME and then set the corresponding long filename through the command CMD_SET_LONG_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the long filename directory will be created successfully.

1.27. CMD_AUTO_DEMO

This command is used for automatic chip demonstration function. CH378 cannot continue to receive the

command codes during the automatic demonstration. The whole automatic demonstration steps are as follows:

① RDY# pin of the chip outputs high level, and automatic demonstration is began;
② CH378 performs various internal initializations;
③ Check whether SD card or USB storage device is inserted, and give priority to SD card;
④ Initialize SD card or USB storage device. If initialization fails, go to ⑥;
⑤ Create a new file named "Chip Demo.TXT" in the root directory of SD card or USB storage device, and write the current chip information (chip version, communication interface mode, plug-in device, file system format, sector size, total capacity, residual capacity, etc.);
⑥ Chip automatic demonstration ends. If automatic demonstration succeeds, RDY# pin will output low level.

After the command CMD_CHECK_EXIST detects that the communication connection is normal, the automatic demonstration command can be sent. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the automatic demonstration will be successful. After the automatic demonstration ends, it is suggested that MCU send the command CMD_RESET_ALL or reset CH378 chip through RSTI pin, and then carry out other operations.

1.28. CMD_GET_SHORT_FILE_NAME

This command is used to get the corresponding short filename and path from the file path of the short filename and the full long filename. Before this command is initiated, it is necessary to set the pathname of the short filename through the command CMD_SET_FILE_NAME and then set the corresponding long filename through the command CMD_SET_LONG_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt state is ERR_SUCCESS, the short filename and path will be gotten successfully, and the data can be gotten through the command CMD_RD_HOST_REQ_DATA. The output data includes 2-byte short filename total length (max. 300 bytes) and subsequent short filename characters in sequence.

1.29. CMD_LONG_FILE_OPEN

This command is used to open the corresponding file from the file path of the short filename and the full long filename. Before this command is initiated, it is necessary to set the pathname of the short filename through the command CMD_SET_FILE_NAME and then set the corresponding long filename through the command CMD_SET_LONG_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the file will be opened successfully.

1.30. CMD_LONG_FILE_ERASE

This command is used to delete the corresponding file from the file path of the short filename and the full long filename. Before this command is initiated, it is necessary to set the pathname of the short filename through the command CMD_SET_FILE_NAME and then set the corresponding long filename through the command CMD_SET_LONG_FILE_NAME. CH378 requests an interrupt from MCU after the command is executed. If the interrupt status is ERR_SUCCESS, the file will be deleted successfully.

1.31. CMD_SET_USB_ID

This command is used to set the USB Vendor ID and product ID. This command requires to input four data, which are the low 8 bits of VID, the high 8 bits of VID, the low 8 bits of PID, and the high 8 bits of PID in sequence. If ID is required to be set, this command must be executed before the command SET_USB_MODE.

1.32. CMD_SET_USB_SPEED

This command is used to set the transmission speed of the current USB device. It is necessary to input 1 byte

of USB speed code. 0x00 indicates full speed, 0x02 indicates high speed, and other values are invalid. If the USB transmission speed is required to be set, this command must be executed before the command SET_USB_MODE, otherwise it is configured as a high-speed device by default.

1.33. CMD_GET_USB_SPEED

This command is used to get the transmission speed of the current USB device and return 1 byte of data, which is the actual speed code of the current USB device. 0x00 indicates full speed, 0x02 indicates high speed, and other values are invalid. In the external firmware mode, this command shall be sent when the device descriptor is obtained, so as to reconfigure information such as endpoint size based on the actual speed.

1.34. CMD_INIT_ENDPx

This command is used to initialize the USB endpoint and is used only in the external firmware mode. This command requires to input 7 bytes of data, which are endpoint index, endpoint number, endpoint type, endpoint direction, endpoint package size high byte, endpoint package size low byte and interrupt status code in sequence. If the endpoint is required to be initialized, this command must be executed before the command SET_USB_MODE.

In the external firmware mode, the four default USB endpoints of CH378 are respectively configured as: endpoint 1 interrupt upload endpoint (64 bytes), endpoint 1 batch upload endpoint (512 bytes), endpoint 2 batch upload endpoint (512 bytes), and endpoint 2 batch upload endpoint (512 bytes). The four endpoints can be configured as interrupt or batch endpoints, upload or download endpoints according to the actual needs. Namely, they can be configured as four download endpoints or four upload endpoints to the maximum. If the endpoint is required to be initialized, all four endpoints must be initialized simultaneously.

1.35. CMD_SET_INDEXx_IN

This command is used to set the working mode of IN endpoint. It is necessary to input 2 bytes of data, which are endpoint index and working mode in sequence. For example, if the USB device does not support the USB standard request SET_INTERFACE, the transaction response mode of the transmitter at the endpoint 0 can be set through this command upon the receipt of the request, so as to return STALL to IN transaction. The byte in the corresponding working mode byte is 0x40. Typically, this command is completed within 2uS.

Working mode of IN endpoint

Working mode byte	Name	Bit description of working mode
Bit 7	Reset Synchronization Bit	If the bit is 1, reset the synchronization bit of the endpoint
Bit 6	STALL setting bit	If the bit is 1, set the endpoint response mode to STALL
Bit 5	STALL clearing bit	If the bit is 1, clear STALL feature before the endpoint
Bits 4-0	Invalid	When no data is uploaded in other cases, automatically answer NAK

1.36. CMD_SET_INDEXx_OUT

This command is used to set the working mode of OUT endpoint. It is necessary to input 2 bytes of data, which are endpoint index and working mode in sequence. For example, if the endpoint 1 receiver has an error, the transaction response mode of the receiver at the endpoint 1 can be set through this command, so as to return STALL to OUT transaction. The byte in the corresponding working mode byte is 0x40. Typically, this command is completed within 2uS.

Working mode of OUT endpoint

Working mode byte	Name	Bit description of working mode
Bit 7	Reset Synchronization Bit	If the bit is 1, reset the synchronization bit of the endpoint
Bit 6	STALL setting bit	If the bit is 1, set the endpoint response mode to STALL
Bit 5	STALL clearing bit	If the bit is 1, clear STALL feature before the endpoint
Bits 4-0	Invalid	When data download is received in other cases, automatically answer ACK

1.37. CMD_UNLOCK_USB

This command is used to release the current USB download endpoint buffer, requiring the input of 1 byte of endpoint index. After successfully receiving data packets downloaded by the USB host, CH378 download endpoint firstly locks the current buffer to prevent buffer from being covered before requesting an interrupt from MCU, suspends the USB communication of the endpoint until MCU releases the current endpoint buffer through the command UNLOCK_USB or only after reading the data through the command RD_USB_DATA. The command cannot be executed more or executed less.

1.38. CMD_RD_USB_DATA

This command is used to read the data block from the endpoint buffer of the current USB interrupt and release the current buffer. This command requires to input 1 byte of endpoint index. The output data read firstly is the data block length, namely, the number of bytes of the subsequent data stream, 2 bytes (the low bytes are in front), with valid values ranging from 0 to 512. If the length is not 0, MCU must read the subsequent data from CH378 one by one. After the data is read, CH378 automatically releases the current USB endpoint buffer, so that it can continue to receive data from the USB host.

1.39. CMD_WR_USB_DATA

This command is used to write the data block to the send buffer of the specified USB endpoint and send the data. This command requires to input several bytes of data. The input data written firstly is the endpoint index, followed by the data length, namely, the number of bytes of the subsequent data stream, 2 bytes (the low bytes are in front), with valid values ranging from 0 to 512. If the length is not 0, MCU must write the subsequent data to CH378 one by one.

1.40. CMD_WR_USB_DATA0

This command is used to write the data block to the send buffer of the USB control endpoint and send the data. This command requires to input several bytes of data. The input data written firstly is the endpoint index, followed by the data length, namely, the number of bytes of the subsequent data stream, 1 byte, with valid values ranging from 0 to 64. If the length is not 0, MCU must write the subsequent data to CH378 one by one.

2. USB Device Mode

CH378 has a built-in underlying protocol in USB communication, and has a convenient built-in firmware mode and a flexible external firmware mode. In the built-in firmware mode, CH378 automatically processes all transactions of the default endpoint 0 (control endpoint). The local MCU is only responsible for data exchange, so MCU program is very simple. In the external firmware mode, the external MCU processes various USB requests according to needs, so as to realize the devices conforming to various USB class

specifications.

CH378 chip integrates PLL frequency multiplier, USB interface SIE, data buffer, command interpreter, general firmware program and other major components. USB interface SIE is used for completion of physical USB data receiving and sending, automatic processing of bit tracking and synchronization, NRZI encoding and decoding, bit stuffing, conversion between parallel data and serial data, CRC data check, transaction handshake, error retry and USB bus status detection, etc. The data buffer is used to buffer data sent and received by USB interface SIE. The command interpreter is used to analyze and automatically process various commands submitted by DSP/MCU automatically. Universal firmware programs are used to automatically process various standard transactions of USB default endpoint 0.

There are 5 physical endpoints in CH378 chip, which are named after the endpoint index, namely index number 0 endpoint, index number 1 endpoint, index number 2 endpoint, index number 3 endpoint and index number 4 endpoint. In the external firmware mode, if the endpoints are not reinitialized by sending CMD_INIT_ENDPx, all endpoints have the same function as built-in firmware.

After CH378 requests an interrupt from MCU, MCU gets the interrupt status through the command CMD_GET_STATUS. The interrupt status is analyzed as follows:

Interrupt status byte	Name	Analysis and description of interrupt status
Bits 7-5	Reserved	Always 000
Bits 4-3	Current transaction	00 = OUT transaction 10 = IN transaction 11 = SETUP transaction
Bits 2-0	Current endpoint	000 = endpoint 0 001 = endpoint 1 010 = endpoint 2 011 = endpoint 3 100 = endpoint 4 101 = USB suspended 110 = USB bus reset

The interrupt status values are described below. In the USB device mode of the built-in firmware mode, MCU only needs to process the interrupt status marked in gray in the table, and CH378 automatically processes other interrupt statuses.

Interrupt status value	Status name	Analysis and description of interrupt causes
0x18	USB_INT_EP0_SETUP	The receiver of the control endpoint receives the data. SETUP succeeds
0x00	USB_INT_EP0_OUT	The receiver of the control endpoint receives the data. OUT succeeds
0x10	USB_INT_EP0_IN	The transmitter of the control endpoint transmits the data. IN succeeds
0x01	USB_INT_INDEX1_OUT	The receiver of the index number 1 endpoint receives the data. OUT succeeds
0x11	USB_INT_INDEX1_IN	The transmitter of the index number 1 endpoint transmits the data. IN succeeds
0x02	USB_INT_INDEX2_OUT	The receiver of the index number 2 endpoint receives the data. OUT succeeds
0x12	USB_INT_INDEX2_IN	The transmitter of the index number 2

		endpoint transmits the data. IN succeeds
0x03	USB_INT_INDEX3_OUT	The receiver of the index number 3 endpoint receives the data. OUT succeeds
0x13	USB_INT_INDEX3_IN	The transmitter of the index number 3 endpoint transmits the data. IN succeeds
0x04	USB_INT_INDEX4_OUT	The receiver of the index number 4 endpoint receives the data. OUT succeeds
0x14	USB_INT_INDEX4_IN	The transmitter of the index number 4 endpoint transmits the data. IN succeeds
0x05	USB_INT_BUS_SUSP	USB bus suspended event
0x06	USB_INT_BUS_RESET	USB bus reset detected
0x07	USB_INT_SET_CONFIG	USB device receives the command SET_CONFIG

Note: the chip will automatically clear USB device interrupts USB_INT_BUS_SUSP, USB_INT_BUS_RESET and USB_INT_SET_CONFIG after notifying the external system.

2.1. Description of Internal Firmware

2.1.1. Index Number 0 Endpoint

The index number 0 endpoint is the control endpoint, which is bidirectional, supports upload and download and has the maximum packet size of 64 bytes.

After successfully completing the SETUP transaction, CH378 will automatically set the synchronous trigger flag of the receiver and transmitter for the endpoint to 1, and automatically process SETUP package.

When successfully completing the OUT transaction, CH378 will automatically trigger the synchronous trigger flag of the receiver for the endpoint from 0 to 1 and from 1 to 0.

When successfully completing the IN transaction, CH378 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint from 0 to 1 and from 1 to 0.

2.1.2. Index Number 1 Endpoint

This endpoint is the interrupt upload endpoint with the endpoint number of 0x81 and the maximum packet size of 64 bytes.

When successfully completing the IN transaction of the endpoint, CH378 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint from 0 to 1 and from 1 to 0. Then the external MCU is notified by using USB_INT_INDEX1_IN as the interrupt status.

2.1.3. Index Number 2 Endpoint

This endpoint is the batch download endpoint with the endpoint number of 0x01 and the maximum packet size of 512 bytes.

When successfully completing the OUT transaction of the endpoint, CH378 will automatically trigger the synchronous trigger flag of the receiver for the endpoint from 0 to 1 and from 1 to 0. Then the external MCU is notified by using USB_INT_INDEX2_OUT as the interrupt status.

2.1.4. Index Number 3 Endpoint

This endpoint is the batch upload endpoint with the endpoint number of 0x82 and the maximum packet size of 512 bytes.

When successfully completing the IN transaction of the endpoint, CH378 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint from 0 to 1 and from 1 to 0. Then the external

MCU is notified by using USB_INT_INDEX3_IN as the interrupt status.

2.1.5. Index Number 4 Endpoint

This endpoint is the batch download endpoint with the endpoint number of 0x02 and the maximum packet size of 512 bytes.

When successfully completing the OUT transaction of the endpoint, CH378 will automatically trigger the synchronous trigger flag of the receiver for the endpoint from 0 to 1 and from 1 to 0. Then the external MCU is notified by using USB_INT_INDEX4_OUT as the interrupt status.

2.1.6. Reference Flow

MCU program in internal firmware mode is provided in the CH378 evaluation board data. The following procedure is used for the external MCU to handle the request for reference.

(I) After MCU is started, it first initializes CH378 as a USB device mode using internal firmware, and then sets the interrupt.

(II) When receiving the interrupt, MCU uses the command GET_STATUS to get the interrupt status. Analysis and processing are as follows:

(1) If the IN transaction of the index number 1 endpoint or index number 3 endpoint is successful, the main program will be notified to continue, for example, write the next packet of data for uploading through the command CMD_WR_USB_DATA.

(2) If OUT transaction of the index number 2 endpoint or index number 4 endpoint is successful, the command CMD_RD_USB_DATA will be used to read the data and notify the main program to process it.

(3) If USB bus is reset, CH378 will clear the USB address and the synchronous trigger flag, etc.

2.2. Description of External Firmware

2.2.1. Index Number 0 Endpoint

The index number 0 endpoint is the control endpoint, which is bidirectional, supports upload and download and has the maximum packet size of 64 bytes.

After successfully completing the SETUP transaction, CH378 will automatically set the synchronous trigger flag of the receiver and transmitter for the endpoint to 1, Then the external MCU is notified by using USB_INT_EP0_SETUP as the interrupt status to read and process SETUP data.

When successfully completing the OUT transaction, CH378 will automatically trigger the synchronous trigger flag of the receiver for the endpoint from 0 to 1 and from 1 to 0. Then the external MCU is notified to read and process the data by using USB_INT_EP0_OUT as the interrupt status.

When successfully completing the IN transaction, CH378 will automatically trigger the synchronous trigger flag of the transmitter for the endpoint from 0 to 1 and from 1 to 0. Then the external MCU is notified to continue to process by using USB_INT_EP0_IN as the interrupt status.

2.2.2. Index Number 1 Endpoint

The endpoint is a unidirectional endpoint and can be configured as either upload or download endpoint. The endpoint number is valid from 1 to 8, and the maximum packet size of the endpoint is 512 bytes.

The endpoint can be configured flexibly according to actual needs through the command CMD_INIT_ENDPx. If it is not configured, it will be interrupt upload endpoint by default. The endpoint number is 0x81, the maximum packet size of the endpoint is 64 bytes, and the generated interrupt status is USB_INT_INDEX1_IN.

2.2.3. Index Number 2 Endpoint

The endpoint is a unidirectional endpoint and can be configured as either upload or download endpoint. The endpoint number is valid from 1 to 8, and the maximum packet size of the endpoint is 512 bytes.

The endpoint can be configured flexibly according to actual needs through the command CMD_INIT_ENDPx. If it is not configured, it will be batch download endpoint by default. The endpoint number is 0x01, the maximum packet size of the endpoint is 512 bytes, and the generated interrupt status is USB_INT_INDEX2_OUT.

2.2.4. Index Number 3 Endpoint

The endpoint is a unidirectional endpoint and can be configured as either upload or download endpoint. The endpoint number is valid from 1 to 8, and the maximum packet size of the endpoint is 512 bytes.

The endpoint can be configured flexibly according to actual needs through the command CMD_INIT_ENDPx. If it is not configured, it will be batch upload endpoint by default. The endpoint number is 0x82, the maximum packet size of the endpoint is 512 bytes, and the generated interrupt status is USB_INT_INDEX3_IN.

2.2.5. Index Number 4 Endpoint

The endpoint is a unidirectional endpoint and can be configured as either upload or download endpoint. The endpoint number is valid from 1 to 8, and the maximum packet size of the endpoint is 512 bytes.

The endpoint can be configured flexibly according to actual needs through the command CMD_INIT_ENDPx. If it is not configured, it will be interrupt upload endpoint by default. The endpoint number is 0x02, the maximum packet size of the endpoint is 512 bytes, and the generated interrupt status is USB_INT_INDEX4_OUT.

2.2.6. Reference Flow

MCU program in external firmware mode is provided in the CH378 evaluation board data. The following procedure is used for the external MCU to handle the request for reference.

(I) After MCU is started, firstly initialize CH378 as a USB device mode using external firmware. According to needs, the four index endpoints of CH378 can be reconfigured. This operation must be done before the command SET_USB_MODE is sent, and the four index endpoints must be configured at the same time, and then the interrupt must be set.

(II) When receiving the interrupt, MCU uses the command GET_STATUS to get the interrupt status. Analysis and processing are as follows:

(1) If the IN transaction of the index number x endpoint (the valid values are 1-4) is successful, the main program will be notified to continue, for example, write the next packet of data for uploading through the command CMD_WR_USB_DATA.

(2) If OUT transaction of the index number x endpoint (the valid values are 1-4) is successful, the command CMD_RD_USB_DATA will be used to read the data and notify the main program to process it.

(3) If the SETUP transaction of the index number 0 endpoint (control endpoint) is successful, the command CMD_RD_USB_DATA will be used to read the data. Analysis and processing are as follows.

(a) If it is a USB request CLEAR_FETURE, analyze and process according to FETURE in the request and the endpoint number. For requests that need to return STALL, set through the command SET_INDEXx_IN.

(b) If it is a USB request GET_DESCRIPTOR, use the command CMD_WR_USB_DATA0 to return all descriptors or the first 64 bytes of the descriptors, and save the USB request and the current descriptor count for subsequent return.

(c) If it is a USB request SET_ADDRESS, save the USB address value set by the host. No

other processing is required. This command is automatically processed by CH378 chip.

(d) If it is a USB request SET_CONFIG, save the set value, and notify the main program whether the USB initialization is successful or not.

(e) If it is a USB request GET_CONFIG, use the command CMD_WR_USB_DATA0 to return the current configuration value.

(f) If it is a USB request GET_INTERFACE, use the command CMD_WR_USB_DATA0 to return the current interface value.

(g) If it is a USB request GET_STATUS, use the command CMD_WR_USB_DATA0 to return the current status value.

(h) Other USB requests are handled as needed. If it is not supported, use the command SET_INDEXx_IN to set the response to STALL.

(4) If the OUT transaction of the index number 0 endpoint (control endpoint) is successful, the command CMD_RD_USB_DATA will be used to read the data. The data can be discarded.

(5) If the IN transaction of the index number 0 endpoint (control endpoint) is successful, and the previous USB request is GET_DESCRIPTOR, the command CMD_WR_USB_DATA0 will be used to return the residual descriptors.

(6) If USB bus is reset, CH378 will clear the USB address and the synchronous trigger flag, etc.